NPS55HH73041A

NAVAL POSTGRADUATE SCHOOL

Monterey, California

A NEW METHOD FOR GLOBAL OPTIMIZATION

by

James K. Hartman

April 1973

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral M. B. Freeman, USN
Superintendent

M. U. Clauser
Provost

ABSTRACT

The basic descent algorithms for minimizing nonlinear objective functions will generally find a local minimum. For problems with multimodal objective functions, it is desirable to extend the search in an attempt to find a global minimum. Several versions of a new method for doing this are presented. Computational tests are performed to compare these methods with each other and with existing methods.

Prepared by:

A New Method for Global Optimization

I. Introduction

The basic descent algorithms for minimizing nonlinear objective functions will generally find a local minimum. For problems with multimodal objective functions, it is desirable to extend the search in an attempt to find a global minimum. Some approaches to this formidable problem have been surveyed by McCormick [1].

A simple strategy for avoiding local solutions to multimodal problems which has been suggested frequently is to repeat the local optimization process several times starting from different initial points x^k , k=1,2,... . If all of the optimizations converge to the same solution x^* , then the problem is probably unimodal, and we have more confidence in the solution x^* than if only one minimization had been performed. If several different local solutions are found by the repeated minimizations, then we are warned that the problem is multimodal, and the chance that the global solution has been found is greater than if only one minimization had been performed.

Some strategies for choosing the starting points for the successive optimizations to improve the chances of finding the global solution were described and compared by Hartman [2]. In this paper a new method which emphasizes the intelligent selection of starting points is developed and evaluated by computational tests.

II. The Problem

We concentrate on the "essentially unconstrained" problem

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to} \quad x \in S \subseteq E^n \end{aligned} \tag{1}$$

where S is a bounded subset of E^n and where $f(x)$ is a nonlinear function which attains its (possibly numerous) local minima in the interior of S. Thus, the set S serves only as a bounded region within which starting points for successive unconstrained minimizations are chosen. A convenient choice for S is the rectangular region

$$S = \{ x \in E^n \mid L_j \leq x_j \leq U_j \ , \ j=1,\ldots,n \} \tag{2}$$

determined by imposing upper and lower bounds on each of the variables x_j. The algorithm of this paper will be developed for rectangular S; the basic idea of the method could be extended to more complex sets S at the cost of additional bookkeeping in the algorithm. We also assume throughout that the function $f(x)$ satisfies any continuity and differentiability assumptions required by the unconstrained minimization method to be used.

III. Development of the Method

Probably the simplest method for selecting the initial point $x^k \in S$ for the k^{th} minimization ($k=1,2,\ldots$) is to choose x^k randomly in S. A random choice is not very efficient, however, since it ignores all the information which has been gathered in the k-1 minimizations already completed. The method of this paper is based on the idea that if the starting points for successive minimizations are chosen to avoid re-searching territory which has already been searched during a previous minimization, then a more efficient procedure will result.

Implementation of this idea will involve the following four steps:

1. Develop a mathematical description of "the territory which has been searched".

2. Select the starting point $x^k \in S$ for the k^{th} minimization to be "as far as possible" from the territory which has been searched by the first $k-1$ minimizations. Thus we attempt to ensure that each successive minimization covers some new territory.

3. As the k^{th} minimization proceeds, modify the description to include the new territory being searched.

4. Terminate the k^{th} minimization prior to convergence if it enters territory which has already been searched. Thus we avoid repeatedly searching to the same local minimum.

The details of these four steps will determine a particular search algorithm. In this paper we will describe three algorithms A1, A2, and A3 based on the above outline, and evaluate them by computational experiments.

Step 1. The "territory which has been searched" is described by partitioning the set S into disjoint cells S_i, $i=1,\ldots,N$. Then a list of the cells

which have been "searched" (see Step 3) describes the subset of S which we call "the territory which has been searched."

We assume that the cells are such that at most one local minimum is contained in any one cell and that if a cell has been searched and a local minimum not found, then there is no minimum in that cell. For all the tests described in this paper, we have restricted our attention to rectangular regions S as in (2). For such regions an easily described partition is given by partitioning the j^{th} coordinate interval $[L_j,U_j]$ into m_j equal subintervals $(j=1,\ldots,n)$. Then there are $N = \prod_{j=1}^{n} m_j$ cells. We have used this partitioning scheme throughout. Other sets S and other partitionings might involve considerably more bookkeeping, but the idea of the method would remain the same.

Step 2. In order to select the starting point x^k for the k^{th} minimization as far as possible from the cells on the list of those which have been searched, we need a measure of the distance $d(x,c^i)$ from an arbitrary point x to the center c^i of the i^{th} cell S_i. Two distance functions were investigated.

$$\text{a)}\quad d(x,c^i) = \left(\sum_{j=1}^{n} (x_j-c_j^i)^2\right)^{1/2} \tag{3}$$

the Euclidean distance function, and

$$\text{b)}\quad d(x,c^i) = \max \{|x_j-c_j^i| \ , \ j=1,\ldots,n \} \tag{4}$$

Then defining the distance from x to the territory already searched to be

$$D(x) = \min \{ d(x,c^i) \mid S_i \text{ is on the list of cells already searched} \} \tag{5}$$

we want to choose x^k to satisfy

$$D(x^k) = \max_{x \in S} D(x) \tag{6}$$

that is, choose x^k to be as far as possible from the closest cell center which has already been searched.

This could be formulated as a deterministic optimization problem — for distance function b a linear program would result - but it was felt that the effort of solving such a program would be excessive. Instead, for purposes of our algorithms, the following approximate heuristic was adopted: Choose L points $x \in S$ at random, evaluate $D(x)$ for each, and let the next starting point x^k be the one of these L points with the largest value of $D(x)$. For our computations $L = 25$ was used throughout.

In the computational tests performed, no consistent superiority of either distance function was observed. For the algorithms of this paper, we will concentrate on the Euclidean distance function (3).

Step 3. As the k^{th} minimization proceeds, the cells which are searched by this minimization must be added to the list to represent the additional territory covered by this search. For the algorithms of this paper, a cell is considered to have been "searched" if the unconstrained minimizer passed through it. In particular, most descent methods for unconstrained minimization involve a succession of linear searches. For our algorithms, the cell which contains the endpoint of each linear search is added to the list of cells (if it is not already on the list). In the early stages of the algorithm development, this choice was contrasted with both more and less frequent additions to the list. The methods were found not to be sensitive to small changes in the frequency of addition to the list, so this simple choice which worked as well as any was selected.

Step 4. The three different algorithms developed in this paper differ in the extent to which the successive minimizations are completed. In algorithm A1 , each unconstrained minimization is continued until a local minimum is found. In algorithm A2 , the k^{th} minimization is terminated if it enters a cell which has already been found to contain a local minimum at some previous search. Algorithm A3 stops a minimization if it enters a cell which was searched (placed on list) by any previous minimization.

Algorithms A2 and A3 attempt to avoid re-searching cells which have already been searched, and hence should be more efficient.

For purposes of comparison, we will also consider the simple algorithm A0 in which all minimizations are completed and in which the starting points x^k are selected at random in S. Thus for A0,

$$x_j^k = L_j + \xi_j (U_j - L_j) \quad j=1,\ldots,n$$

where each ξ_j is a uniform random variable on the interval [0,1]. This algorithm makes no use whatever of information from the previous minimizations.

IV. Computational Tests.

Computational experiments to compare the four methods A0, A1, A2, and A3 were performed utilizing the facilities of the W. R. Church Computer Center at the Naval Postgraduate School. For each of the test functions employed, each algorithm was run 30 times with different random number sequences; the average of these 30 runs is reported here. A run was allowed to continue until the algorithm had required 1000 evaluations of the objective function $f(x)$.

The same test problems were used for this study as in Hartman [2]. These problems, with predictable local and global solutions were constructed using the objective function

$$f(x) = - \sum_{i=1}^{m} c_i \exp[(x-p_i)' A_i (x-p_i)]$$

This function consists of the superposition of m modes, where mode i has depth $c_i \in E^1$, position $p_i \in E^n$, and shape and width determined by the $n \times n$ negative definite matrix A_i. Particular test functions were obtained by choosing c_i and p_i from a random number table. A_i was then chosen to ensure that the m modes were narrow enough that they did not completely merge into one another.

Each of the algorithms requires an unconstrained minimization method. To establish comparability with an earlier study (Hartman [2]) Powell's derivative free method [3] was selected.

V. Results

The results of the computational tests performed are summarized in Tables 1 and 2. Table 1 gives characteristics of the 10 test problems used. Table 2 lists, for each problem and each algorithm, the average (over 30 replications) best f value obtained after 250, 500, 750, and 1000 function evaluations. A * indicates that all 30 trials achieved the global minimum. The percentage of the 30 trials which did not locate the global minimum after 1000 function evaluations is also given in Table 2.

Comparison of A0 and A1 shows the advantage of using the available information in selecting starting points over random selection (this is the only difference between A0 and A1). A1 performed better than A0 on 9 of the 10 test functions. Further comparison of A1 with A2 and A3 shows the incremental advantage of early termination of minimizations which are not covering new ground. A3 tends to terminate these minimizations more quickly than A2 , and as might be expected, it generally performs better. The two variable problems A - E were handled very well by A3 ; all 30 replications attained the global minimum by the 500th function evaluation for each of these functions. The five variable problems proved considerably harder for all four methods, but the methods which utilize past information usually did better than the random method A0.

Although the algorithms presented here are generally successful, there is clearly a need for more work to improve these methods and to develop entirely new methods for global optimization.

Problem	Variables	Number of Minima	Value of Global Minimum
A	2	4	- 9.0
B	2	10	- 9.9
C	2	10	- 9.3
D	2	10	- 9.8
E	2	10	-13.0
F	5	5	- 9.4
G	5	5	-10.1
H	5	10	-10.0
I	5	10	- 8.9
J	5	20	-11.9

Table 1

Characteristics of Test Problems

Function		A0	A1	A2	A3
A	best f after 250	- 8.7	- 9.0	*	*
	best f after 500	- 8.9	*	*	*
	best f after 750	*	*	*	*
	best f after 1000	*	*	*	*
	% failures	0.0	0.0	0.0	0.0
B	best f after 250	- 9.1	- 9.3	- 9.4	- 9.6
	best f after 500	- 9.3	- 9.9	*	*
	best f after 750	- 9.8	*	*	*
	best f after 1000	- 9.9	*	*	*
	% failures	3.3	0.0	0.0	0.0
C	best f after 250	- 8.4	- 8.7	- 9.1	- 9.1
	best f after 500	- 8.6	- 8.9	*	*
	best f after 750	- 8.8	- 9.2	*	*
	best f after 1000	- 9.0	*	*	*
	% failures	26.7	0.0	0.0	0.0
D	best f after 250	- 9.3	- 9.3	- 9.6	- 9.6
	best f after 500	- 9.6	- 9.8	- 9.8	*
	best f after 750	- 9.7	- 9.8	*	*
	best f after 1000	- 9.8	- 9.8	*	*
	% failures	10.0	3.3	0.0	0.0
E	best f after 250	-10.8	-12.3	-12.6	-12.4
	best f after 500	-12.1	-12.9	-12.8	-13.0
	best f after 750	-12.4	-13.0	-12.9	*
	best f after 1000	-12.7	*	-13.0	*
	% failures	10.0	0.0	3.3	0.0
F	best f after 250	- 7.0	- 7.3	- 7.0	- 7.7
	best f after 500	- 8.0	- 8.3	- 8.7	- 8.5
	best f after 750	- 8.6	- 8.6	- 9.0	- 8.9
	best f after 1000	- 8.7	- 8.8	- 9.2	- 9.0
	% failures	60.0	56.7	26.6	43.3
G	best f after 250	- 7.9	- 8.9	- 9.2	- 9.5
	best f after 500	- 9.4	- 9.9	-10.0	-10.0
	best f after 750	- 9.8	-10.1	-10.1	-10.1
	best f after 1000	-10.1	-10.1	*	*
	% failures	3.3	3.3	0.0	0.0

* all 30 trials attained global minimum.

Table 2

Test Results

Function		A0	A1	A2	A3
H	best f after 250	- 7.9	- 8.2	- 8.0	- 7.9
	best f after 500	- 8.4	- 8.7	- 8.7	- 8.7
	best f after 750	- 8.7	- 8.7	- 9.1	- 9.2
	best f after 1000	- 9.0	- 9.1	- 9.3	- 9.4
	% failures	76.7	70.0	63.3	53.3
I	best f after 250	- 7.7	- 8.1	- 7.7	- 7.6
	best f after 500	- 8.0	- 8.6	- 8.5	- 8.4
	best f after 750	- 8.7	- 8.7	- 8.8	- 8.7
	best f after 1000	- 8.9	- 8.8	- 8.8	- 8.9
	% failures	10.0	16.7	13.3	10.0
J	best f after 250	- 7.6	- 6.9	- 7.2	- 7.1
	best f after 500	- 8.8	- 8.0	- 8.1	- 8.0
	best f after 750	- 9.2	- 9.6	- 9.2	- 8.8
	best f after 1000	- 9.8	- 9.9	- 9.9	-10.1
	% failures	43.3	40.0	43.3	33.3

* all 30 trials attained global minimum

Table 2 (continued)

Test Results

References

[1] McCormick, G. P., "Attempts to Calculate Global Solutions of Problems That May Have Local Minima", George Washington University, Report T-253, December 1971.

[2] Hartman, J. K., "Some Experiments in Global Optimization", Naval Postgraduate School, Technical Report NPS55HH72051A, May 1972.

[3] Powell, M. J. D., "An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives", Computer Journal, Vol. 7, No. 2, p. 155, 1964.

INITIAL DISTRIBUTION LIST

	No. Copies
Defense Documentation Center Cameron Station Alexandria, Virginia 22314	12
Dean of Research Code 023 Naval Postgraduate School Monterey, California 93940	1
Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
Dr. Thomas Varley Office of Naval Research Arlington, Virginia 22217	1
Library, Code 55 Department of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	5
Professor Donald P. Gaver Department of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	1
Professor James K. Hartman Department of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	10

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author)

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

REPORT TITLE

A New Method for Global Optimization

DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Technical Report

AUTHOR(S) (First name, middle initial, last name)

James K. Hartman

REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
April 1973	15	3

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	NPS55HH73041A
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

The basic descent algorithms for minimizing nonlinear objective functions will generally find a local minimum. For problems with multimodal objective functions, it is desirable to extend the search in an attempt to find a global minimum. Several versions of a new method for doing this are presented. Computational tests are performed to compare these methods with each other and with existing methods.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Nonlinear Programming						
Mathematical Programming						
Optimization						
Global Optimization						
Search Methods						